**In the Specification:**

Please amend the paragraph beginning on page3, line 8 as follows:

A number of preferred embodiments of the present invention will now be described with reference to Fig. 1, in which is a flow diagram of a method of determining whether an algebraic expression is syntactically correct. Fig. 2 is a flow diagram describing a step of Fig. 1 in which a character string is created from an algebraic expression being analyzed.

Please amend the paragraphs beginning on page 5, line 4 and ending on page 6, line3 as follows:

In step 30, the character string `strg` is created from the algebraic expression being analyzed. Thus for a given character string, the derived character string `strg` obtained by step 30 will be a sequence of delimiter characters and type characters. Step 30 is performed by ~~the following~~ [[sub]]steps 31-38 in the flow diagram of FIG. 2. :

In [[sub]]step 31 ~~(not shown)~~, it is determined whether the given algebraic expression begins with a unitary + or – operator. If this is so, then the expression is modified in step 32 by being prefixed with the numeral `0`. Alternatively, the algebraic expression may be enclosed in brackets in step 32. Furthermore, all blank (or space) characters are removed from the algebraic expression in step 32. A counter variable `i` is initiated to `0` in step 33.

In [[sub]]step [[32]] 34 ~~(not shown)~~, the expression is scanned from left to right, character by character, until a delimiter character is found or the end of the expression is reached.

A variable delimiter character α is set equal to the delimiter character found.

(a) If in step 35 no characters are found before the delimiter character α, then in step 37 the i-th character in the character string `strg` is set equal to the delimiter character as follows: `strg[i] = α.` ~~The~~ Also in step 37, the counter variable `i` is ~~also~~ incremented by 1 and the procedure continues to [[sub]]step ~~33 (not shown)~~38.

(b) If in step 35 one or more characters are found before the delimiter then it is determined in step 36 whether it/they form(s) a valid variable name, function name or number. If the character(s) is/are valid, then a character "v" is put into string position `strg[i]`. Alternatively a character "?' is put into string position `strg[i]`. The following string position in the string `strg[]` is filled with the delimiter character α as follows: `strg[i+1] = α`. The counter variable `i` is increased by 2 and the procedure continues to [[sub]]step [[33]] 38.

~~Substep 33~~ Step 38 determines whether the end of the expression has been reached. If this is so, then the procedure ends. If this is not so, then the procedure continues to [[sub]]step [[32]] 34, assuming that the expression now begins at the character immediately following the delimiter character a.

Please amend the paragraph beginning on page 8, line 11 as follows:

Source or psuedo-code for performing steps 50 to 90 of the method 100 is as follows:

```
// strg[] is the character array derived from the given expression

// using [[sub]]steps 31-[[33]]38 described above.

// size is the size of the string strg.

// n is the dimension of FromStrg[] and ToStrg[] arrays.

// ChangeSubstrg() replaces FromStrg[i], if found in strg with

ToStrg[i].

// cond is a boolean flag which saves the result of the iteration.

// It is TRUE if the strg is syntactically correct, else FALSE.

size = strlen(strg) + 1;

cond = TRUE;

while (cond) {

  if(strchr(strg,(int)'?')break;

  i = -1

  while (++i < n)  {

     while (strstr(strg, FromStrg[i]))

     ChangeSubstrg(strg, FromStrg[i], ToStrg[i]);
```

```
    }

    j = strlen(strg) + 1;

    if (size == j) break;

    size = j;

  }

  if (strcmp(strg, "v") != 0) cond = FALSE;

  if (cond) Msg("Expression is correct");

  else Msg("Expression is incorrect");
```